High-Precision Computation: Mathematical Physics and Dynamics

D. H. Bailey* R. Barrio† J. M. Borwein‡

September 26, 2011

Abstract

At the present time, IEEE 64-bit floating-point arithmetic is sufficiently accurate for most scientific applications. However, for a rapidly growing body of important scientific computing applications, a higher level of numeric precision is required. Such calculations are facilitated by high-precision software packages that include high-level language translation modules to minimize the conversion effort. This paper presents a survey of recent applications of these techniques and provides some analysis of their numerical requirements. These applications include supernova simulations, climate modeling, planetary orbit calculations, Coulomb n-body atomic systems, studies of the fine structure constant, scattering amplitudes of quarks, gluons and bosons, nonlinear oscillator theory, experimental mathematics, evaluation of recurrence relations, numerical integration of ODEs, computation of periodic orbits, studies of the splitting of separatrices, detection of strange nonchaotic attractors, Ising theory, quantum field theory, and discrete dynamical systems. We conclude that high-precision arithmetic facilities are now an indispensable component of a modern large-scale scientific computing environment.

1 Introduction

Virtually all present-day computer systems, from personal computers to the largest supercomputers, implement the IEEE 64-bit floating-point arithmetic standard, which provides 53 mantissa bits, or approximately 16 decimal digit accuracy. For most scientific applications, 64-bit arithmetic is more than sufficient, but for a rapidly expanding body of applications, it is not. In this paper we will examine a variety of applications where high-precision arithmetic is useful:

*Lawrence Berkeley National Laboratory, Berkeley, CA 94720, dhbailey@lbl.gov. Supported in part by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy, under contract number DE-AC02-05CH11231.

†Depto. Matemática Aplicada and IUMA, Universidad de Zaragoza, E-50009 Zaragoza, Spain rbarrio@unizar.es. Supported in part by the Spanish research project MTM2009-10767.

‡Centre for Computer Assisted Research Mathematics and its Applications (CARMA), University of Newcastle, Callaghan, NSW 2308, Australia, jonathan.borwein@newcastle.edu.au. Supported in part by the Australian Research Council.

1. Numerically sensitive calculations (ill-conditioned problems). Some scientific computations include sensitive portions that produce inaccurate results when performed using straightforward algorithms and 64-bit arithmetic. These inaccurate results may in turn induce other errors, such as taking the wrong path in a conditional branch. Often such errors can be overcome by using higher-precision arithmetic in just one or two spots.

2. Long-time simulations. Almost any kind of physical simulation, if performed over many time intervals, will eventually depart from reality, due to cumulative round-off error. High-precision arithmetic can eliminate much of this error, although errors due to the discretization of time and space may remain.

3. Large-scale simulations. Computations that are well-behaved on modest-sized problems, such as those run on a single-CPU system, may exhibit significant numerical errors when scaled up to the huge sizes typical of those now being run on large systems with many Tbytes of memory and well over 100,000 processor cores.

4. Small-scale phenomena. When studying why some behavior appears in a system (or if truly appears at all), it is often necessary to employ a very fine-scale resolution to "zoom" in on the phenomena. These fine-scale computations often require higher-precision arithmetic to fully resolve.

5. "Experimental mathematics" computations. Recent work in experimental mathematics has highlighted the effectiveness of employing extremely high precision (hundreds or even thousands of digits) to uncover new identities and relations. One example is the analytic evaluation of classes of integrals that arise in mathematical physics.

With regards to item 1, it should be kept in mind that the vast majority of persons currently performing numerical computations are not experts in numerical analysis, and this fact is not likely to change anytime soon. For example, in 2010 at the University of California, Berkeley, a total of 219 students enrolled in the two sections of Math 128A, a one-semester introductory numerical analysis course required of applied math majors, but only 24 enrolled in Math 128B, a more advanced course. By contrast, in the same year a total of 870 seniors graduated in the Division of Mathematical and Physical Sciences (including Mathematics, Physics and Statistics), the College of Chemistry and the College of Engineering (including Computer Science), most of whom will do some numerical computation in their career work. If we add to this list graduates in other fields with computational components, such as biology, geology, medicine and social sciences, we conclude that only about 2% of the Berkeley graduates each year who likely will be using computational tools in their career work have advanced training in numerical analysis. There is no reason to believe that this ratio is significantly higher elsewhere.

Thus, for the foreseeable future, almost all technical computing will be performed by persons who have had only basic training in numerical analysis, or none at all. Such persons typically rely on relatively straightforward algorithms and pre-existing, off-the-shelf software, focusing most of their efforts on details specific to their discipline (physics, engineering, psychology, etc.). When numerical difficulties are encountered, they seek a simple and easy-to-implement remedy, instead of attempting wholesale algorithm replacement.

High-precision arithmetic is an attractive option for such users, because even in situations where numerically better behaved algorithms are known in the literature that may resolve a numerical problem, it is often both easier and more reliable to simply increase the precision used for the existing algorithm, using tools such as those described in Section 2. At the very least, using high-precision arithmetic to rectify numerical problems buys some time while a better long-term solution is sought.

1.1 Extra precision versus algorithm changes

The following example illustrates some of the issues involved. Suppose one wishes to recover the integer polynomial that produces the result sequence $(1, 32771, 262217, 885493, 2101313, 4111751, 7124761)$ for integer arguments $(0, 1, \ldots, 6)$. While there are several ways to approach this problem, many scientists and engineers will employ a least-squares scheme, since this is a very familiar tool in scientific data analysis, and efficient library software is readily available. Indeed, this approach is suggested in a widely used reference [81, pg. 44]. In this approach, one constructs the $(n+1) \times (n+1)$ linear system

$$\begin{bmatrix} n+1 & \sum_{k=1}^n x_k & \cdots & \sum_{k=1}^n x_k^n \\ \sum_{k=1}^n x_k & \sum_{k=1}^n x_k^2 & \cdots & \sum_{k=1}^n x_k^{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^n x_k^n & \sum_{k=1} x_k^{n+1} & \cdots & \sum_{k=1}^n x_k^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^n y_k \\ \sum_{k=1}^n x_k y_k \\ \vdots \\ \sum_{k=1}^n x_k^n y_k \end{bmatrix}, \qquad (1)$$

where (x_k) are the integer arguments and (y_k) are the sequence values. Then one solves for $(a_1, a_2, \cdots, a_n)$ using, for example, *LINPACK* [53] or *LAPACK* [52] software.

In the specific problem mentioned above, a double-precision (64-bit) floating-point implementation of the least-squares scheme succeeds in finding the correct polynomial coefficients, which, after rounding to the nearest integer, are $(1, 0, 0, 32769, 0, 0, 1)$, or, in other words, $f(x) = 1 + (2^{15} + 1)x^3 + x^6$. Unfortunately, this scheme fails to find the correct polynomial for a somewhat more difficult problem, namely to find the degree-8 polynomial that generates the 9-long sequence $(1, 1048579, 16777489, 84941299, 268501249, 655751251, 1360635409, 2523398179, 4311748609)$, for integer arguments $(0, 1, \cdots, 8)$. The program finds approximate degree-8 polynomial coefficients, but they are not correct, even after rounding to the nearest integer — too much floating-point round-off error has occurred.

Many numerical analysts will point out here that this approach is not the best scheme for this type of problem, in part because the Vandermonde matrix system (1) is known to be rather unstable (this is also pointed out in [81, pg. 44]). A more effective approach in the cases such as this, where the number of inputs is one greater than the degree, is to employ the Lagrange interpolating polynomial, which, given a set of $n+1$ data points $(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)$, is defined as $L(x) = \sum_{j=0}^n y_j p_j(x)$, where

$$p_j(x) \quad = \prod_{0 \le i \le n,\, i \ne j} \frac{x - x_i}{x_j - x_i}. \qquad (2)$$

In the problem at hand, $x_j = j$ for $0 \le j \le n$. In order to minimize numerical error, one should separately compute the polynomial in the numerator and the factorials in the denominator before

Algorithm	Precision (digits)	Problem degree 6	8	12
Least-squares	16	succeeded	failed	failed
	31	succeeded	succeeded	succeeded
Lagrange	16	succeeded	succeeded	failed
	31	succeeded	succeeded	succeeded
Demmel-Koev	16	succeeded	succeeded	failed

Table 1: Success and failure of various polynomial data fit schemes

performing the division. In this way, the chief source of numerical error is the evaluation of the inner products inherent in the formula $L(x) = \sum_{j=0}^{n} y_j p_j(x)$.

This scheme, implemented with 64-bit IEEE arithmetic, correctly deduces that the 9-long data sequence above is produced by the polynomial $1 + (2^{20} + 1)x^4 + x^8$. However, this scheme fails when given the more challenging 13-long input data vector $(1, 134217731, 8589938753, 97845255883, 549772595201, 2097396156251, 6264239146561, 15804422886323, 35253091827713, 71611233653971, 135217729000001, 240913322581691, 409688091758593)$, which is generated by $1 + (2^{27} + 1)x^6 + x^{12}$.

The state-of-the-art algorithm in this area, as far as the present authors are aware, is a technique due to James Demmel and Plamen Koev [50], which accurately solves "totally positive" systems such as (1), where the determinant of any square submatrix is positive. A *Matlab* implementation of this scheme is available at [72]. We found that this program solves the degree-6 and degree-8 problems mentioned above, but, like the Lagrange polynomial scheme, fails for the degree-12 problem.

However, there is another approach to these problems: simply modify the source code of any reasonably effective solution scheme to invoke higher-precision arithmetic. For example, when we modified our Fortran-90 least-squares scheme to employ double-double precision (approximately 31-digit accuracy), using the QD software mentioned in Section 2, we were able to correctly solve all three problems (degrees 6, 8 and 12). Converting the Lagrange polynomial scheme to use double-double arithmetic was even easier, and the resulting program also solved all three problems without incident. These results are summarized in Table 1. No entry is listed for the Demmel-Koev scheme with 31-digit arithmetic, because we relied on a *Matlab* implementation for which a double-double version is not available, although we have no reason to doubt that it would also succeed.

2 High-precision software

Efficient algorithms are known for performing, to any desired precision, the basic arithmetic operations, square and n-th roots, and most transcendental functions [35, pp. 215–245], [36, pp. 299–318], [37, 38, 39, 44]. Software packages implementing these algorithms have been available since the early days of computing. However, many of these packages have required one to rewrite a scientific application with individual subroutine calls for each arithmetic operation. The difficulty of writing and debugging such code has deterred all but a few computational scientists and mathematicians from using such tools.

In the past 10 years or so, high-precision software packages have been produced that include high-level language interfaces that make such conversions relatively painless. These packages typically utilize custom datatypes and operator overloading features, which are available in languages such as C++ and Fortran-90, to facilitate conversion. Even more advanced high-precision facilities are available in the commercial products *Mathematica* and *Maple*, which incorporate arbitrary-precision arithmetic in a naturally integrated way for a wide range of functions, many more than are typically available from freely available software. These two commercial products also provide facilities to convert existing scientific programs written in other languages, although human intervention is often required.

Here are some high-precision arithmetic software packages that are freely available on the Internet, listed in alphabetical order. The ARPREC, QD and MPFUN90 packages are available from the first author's website: `http://crd.lbl.gov/~dhbailey/mpdist`.

- ARPREC. This package includes routines to perform arithmetic with an arbitrarily high level of precision, including many algebraic and transcendental functions. High-level language interfaces are available for C++ and Fortran-90, supporting real, integer and complex datatypes.

- GMP. This package includes an extensive library of routines to support high-precision integer, rational and floating-point calculations. GMP has been produced by a volunteer effort and is distributed under the GNU license by the Free Software Foundation. It is available at `http://gmplib.org`.

- MPFR. The MPFR library is a C library for multiple-precision floating-point computations with exact rounding, and is based on the GMP multiple-precision library. Additional information is available at `http://www.mpfr.org`.

- MPFR++. This is a high-level C++ interface to MPFR. Additional information is available at `http://perso.ens-lyon.fr/nathalie.revol/software.html`. A similar package is GMPFRXX, available at `http://math.berkeley.edu/~wilken/code/gmpfrxx`.

- MPFUN90. This is similar to ARPREC in user-level functionality, but is written entirely in Fortran-90 and provides a Fortran-90 language interface.

- QD. This package includes routines to perform "double-double" (approx. 31 digits) and "quad-double" (approx. 62 digits) arithmetic. High-level language interfaces are available for C++ and Fortran-90, supporting real, integer and complex datatypes. This software is much faster than using arbitrary precision software when only 31 or 62 digits are required.

Just as an example of the simple case, the QD package, which provides double-double and quad-double arithmetic, is based on the following algorithms for the accurate addition and multiplication of two IEEE 64-bit operands using rounded arithmetic, due to Knuth [71] and Dekker [49]:

function $[x, y] = \mathtt{TwoSum}(a; b)$
$x = \mathtt{fl}(a + b)$
$z = \mathtt{fl}(x - a)$
$y = \mathtt{fl}((a - (x - z)) + (b - z))$

function $[x, y] = \mathtt{Split}(a)$
$c = \mathtt{fl}(\text{factor} \cdot a)$ (in double precision factor $= 2^{27} + 1$)
$x = \mathtt{fl}(c - (c - a))$
$y = \mathtt{fl}(a - x)$

function $[x, y] = \mathtt{TwoProd}(a; b)$
$x = \mathtt{fl}(a \cdot b)$
$[a1,\ a2] = \mathtt{Split}(a)$
$[b1,\ b2] = \mathtt{Split}(b)$
$y = \mathtt{fl}(a2 \cdot b2 - (((x - a1 \cdot b1) - a2 \cdot b1) - a1 \cdot b2))$

In the above, `fl` stands for the floating-point evaluation using rounded arithmetic. These algorithms satisfy the following error bounds [79] (where $\mathbb{F}$ denotes the set of floating-point numbers and u the rounding unit of the computer):

Theorem 1 *For $a, b \in \mathbb{F}$ and $x, y \in \mathbb{F}$,* `TwoSum` *and* `TwoProd` *verify*

$$[x, y] = \mathtt{TwoSum}(a, b), x = \mathrm{fl}(a + b), x + y = a + b, |y| \leq u|x|, |y| \leq u|a + b|,$$
$$[x, y] = \mathtt{TwoProd}(a, b), x = \mathrm{fl}(a \times b), x + y = a \times b, |y| \leq u|x|, |y| \leq u|a \times b|.$$

One downside of using high-precision software is that such facilities greatly increase computer run times, compared with using conventional 64-bit arithmetic. For example, computations using double-double precision arithmetic typically run five to ten times slower than with 64-bit arithmetic. This figure rises to at least 25 times for the quad-double arithmetic, to more than 100 times for 100-digit arithmetic, and to well over 1000 times for 1000-digit arithmetic. However, in many cases, high-precision arithmetic is only needed in one or two places in the code, so that the total run time is not much greater than the standard code. Even when major slowdowns are inevitable, modern highly parallel computer technology often permits such calculations to be completed in reasonable wall-clock run times.

3 Applications of high-precision arithmetic

In this section we give a selection of five directly applied applications:

3.1 Planetary orbit calculations

One central question of planetary theory is whether the solar system is stable over cosmological time frames (many millions or billions of years). Planetary orbits are well known to exhibit chaotic behavior. Indeed, as Isaac Newton once noted, "The orbit of any one planet depends on the combined motions of all the planets, not to mention the actions of all these on each other. To consider simultaneously all these causes of motion and to define these motions by exact laws allowing of convenient calculation exceeds, unless I am mistaken, the forces of the entire human intellect." [56, p. 121].

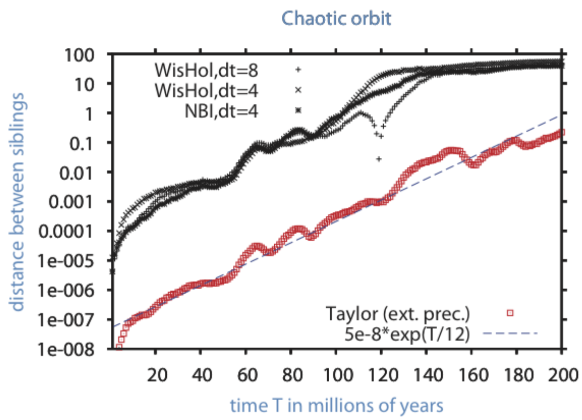

Figure 1: Divergence between nearby trajectories, integrated with four different numerical integrators (the Wisdom-Holman symplectic integrator with two stepsizes, the NBI's 14th order Cowell-Sörmer integrator and the Taylor method to check the results). Left figure: a chaotic trajectory with a Lyapunov time of about 12 million years. Right figure: a trajectory showing no evidence of chaos over 200My. Both trajectories are within observational uncertainty of the outer planetary positions. (Reproduced with permission from [67])

Scientists have studied this question by performing very long-term simulations of planetary motions. These simulations typically do fairly well for long periods, but then fail at certain key junctures, such as when two planets pass fairly close to each other. Researchers have found that double-double or quad-double arithmetic is required to avoid severe numerical inaccuracies, even if other techniques are employed to reduce numerical error [73]. A team led by W. Hayes studied solar system orbits using various numerical ordinary differential equation (ODE) integrators, checked to higher precision using a Taylor series integrator, performed using 19-digit Intel extended precision [67] (see Figure 1). Applegate and others employed a special-purpose computer to investigate the stability of the outer solar system [5]. Note that the use of high-precision in long term integrations is limited to check in short-medium integration times other numerical integrators more suitable for these purposes. In the Figure 1 we observe that all the integrators work well as shown by the close behavior of all the simulations in a case of initial conditions leading to a chaotic or regular movement, and so as Hayes showed, this lead to that these two options are not numerical artefacts, but a real consequence of the uncertainty and the mixing structure of the solar system.

3.2 High-precision solution of ODEs: Taylor method

In several applications of dynamical systems we need to integrate the relevant differential equation, normally for a short time, with very high precision. Moreover, in the study of the bifurcations and stability of periodic orbits (by instance) we also have to integrate the first order variational equations using as initial conditions the identity matrix. To reach this goal we may, obviously, use any numerical ODE method such as Runge-Kutta. During the last few years, the Taylor method has emerged as a preferred method in the computational dynamics community [86].

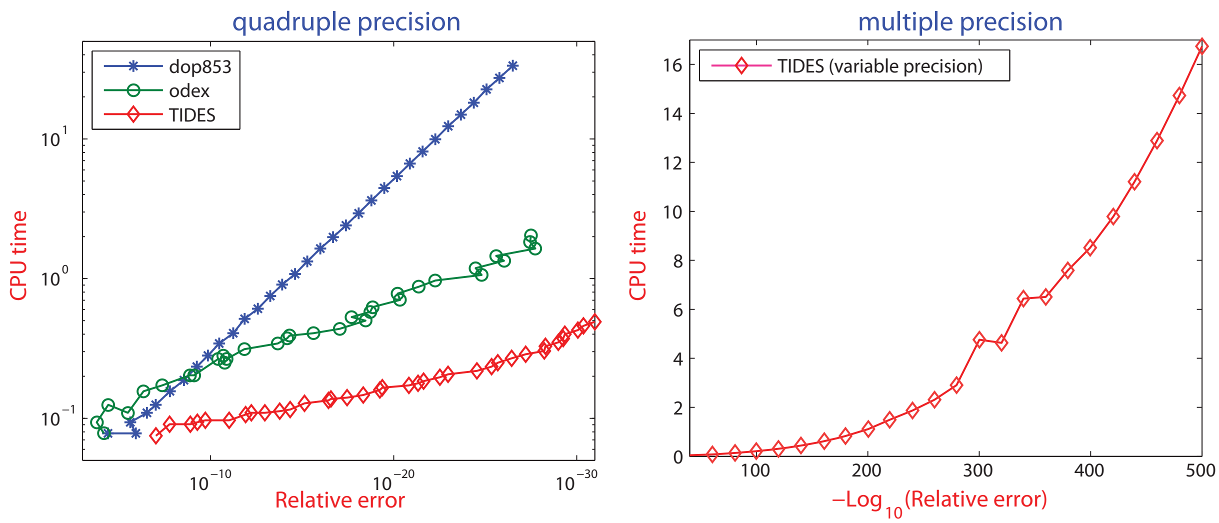

Figure 2: Left: Precision vs. CPU time diagram in quadruple precision for the numerical integration of the unstable periodic orbit LR for the Lorenz model using a Runge-Kutta code (`dop853`), an extrapolation code (`odex`) and a Taylor series method (`TIDES`). Right: Precision vs. CPU time diagram for the multiple-precision numerical integration of an unstable periodic orbit for the Lorenz model using the `TIDES` code.

The Taylor method is one of the oldest numerical methods for solving ordinary differential equations, but it is scarcely used in the numerical analysis community. The formulation is quite simple [22, 25, 47]. Let us consider the initial value problem $\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y})$. Now, the value of the solution at t_i (that is, $\boldsymbol{y}(t_i)$) is approximated by $\boldsymbol{y}_i$ from the n-th degree Taylor series of $\boldsymbol{y}(t)$ at $t = t_i$ (the function $\boldsymbol{f}$ has to be a smooth function). So, denoting $h_i = t_i - t_{i-1}$,

$$\begin{aligned} \boldsymbol{y}(t_0) &=: \boldsymbol{y}_0, \\ \boldsymbol{y}(t_i) &\simeq \boldsymbol{y}_{i-1} + \boldsymbol{f}(t_{i-1}, \boldsymbol{y}_{i-1})\, h_i + \ldots + \frac{1}{n!}\,\frac{d^{n-1}\boldsymbol{f}(t_{i-1}, \boldsymbol{y}_{i-1})}{dt^{n-1}}\, h_i^n \;=:\; \boldsymbol{y}_i. \end{aligned}$$

Therefore, the problem is reduced to the determination of the Taylor coefficients $\{1/(j+1)!\, d^j \boldsymbol{f}/dt^j\}$. This may be done quite efficiently by means of the automatic differentiation (AD) techniques. Note that the Taylor method has several good features (for details see [22, 23, 25]).

In the Fig. 2 we present some comparisons on the Lorenz model [77] for the classical Saltzman's parameter values using the Taylor method (`TIDES` code) and the well established codes `dop853` (a Runge-Kutta code) and `odex` (an extrapolation code) developed by Hairer and Wanner [63]. We observe that in quadruple precision quite soon the Taylor method becomes the fastest and, as expected, the `odex` code is more efficient than the Runge-Kutta code (note that `odex` is a variable order code, as `TIDES`, and so it is more adaptable than the fixed order method). In double precision the most efficient code is the Runge-Kutta code, but for high precision the Taylor series method is the only reliable method among the standard methods. Note that the computer time for a high-precision numerical integration of one period ($T = 1.55865$) of the LR unstable periodic orbit (in symbolic dynamics notation one loop around the left equilibrium point, and one around the right one [90]) maintaining 500 digits is just around 16 seconds using a normal desktop computer, a quite reasonable time.

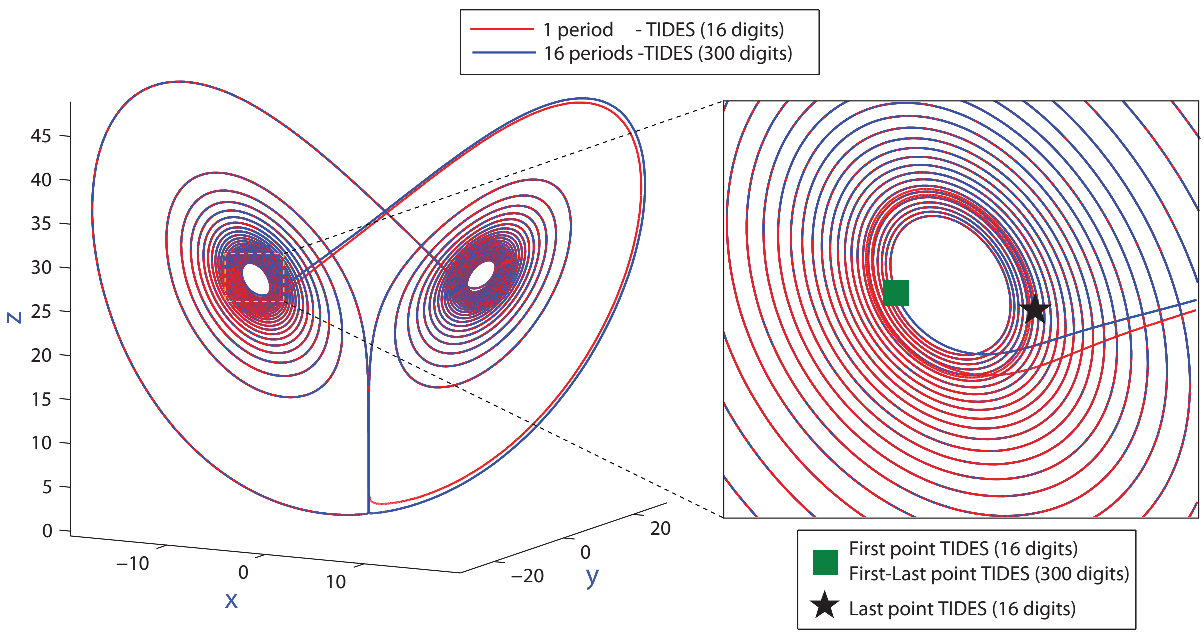

Figure 3: Numerical integration of the $L^{25}R^{25}$ unstable periodic orbit for the Lorenz model during 16 time periods using the `TIDES` code with 300 digits and 1 time periods using double precision.

So, one question is, do we really need such a large accurate numerical integrations in these kind of systems? To illustrate the need we show in Fig. 3 the numerical simulations of 16 time periods using the the `TIDES` code with 300 digits and 1 time period using double precision for the numerical simulation of the $L^{25}R^{25}$ unstable periodic orbit for the Lorenz model. Now we lose more than 16 digits on each period (the period of the orbit is $T = 33.890206423038$ and the largest Lyapunov exponent $\lambda = 0.958$, so $\exp(\lambda T) \approx 1.5324 \cdot 10^{16}$), and therefore it is not possible to simulate any period of this orbit in double precision. The double precision orbit is not periodic (see the zoom) and it also loses the symmetry of the correct orbit. Note that this simulation is among the longest precise numerical calculations presented in the literature for the

Lorenz model, up to a final time $t_f \approx 1150$. Obviously with the code `TIDES` one can go as far as his/her computer is able to compute.

It is important to remark that nowadays there are excellent free-software implementations of the Taylor series method, with arbitrary high-precision, for the numerical solution of ODEs and for the automatic determination of the solution of high-order variational equations. The software `TIDES` [1, 29] (Taylor series Integrator for Differential EquationS) is a powerful implementation of this technology (see `http://gme.unizar.es/software/tides` or send an email to `tides@unizar.es` or `rbarrio@unizar.es`).

3.3 Evaluating recurrence relations

The numerical evaluation of recurrences have the potential of being unstable [58] and in a large number of numerical algorithms we have to use them. A classical example of unstable recurrence is the evaluation of the Bessel function of first kind $J_i(x)$ [3] by means of the three-term recurrence

$$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x). \tag{3}$$

In this case, we miss almost immediately all the significant digits (in fact at $n = 7$) and there is no way of using extended precision to improve the results. This is illustrated in figure 4, where the use of the recurrence gives a growing sequence, when the real result is a fast decreasing one. The reason of this disastrous build-up of errors [58] is due to the fact that the Bessel function of the first kind, but also the Bessel function of the second kind $Y_i(x)$ are solutions of the recurrence relation (3) and $J_i(x)/Y_i(x) \sim (x/2)^{2i}/(2(i!)^2)$ as $i \to \infty$, and so $J_i(x)$ is a (highly) minimal solution at infinity. This implies that any error is extremely amplified and the numerical solution goes quite fast to the dominant one. In this case we have to look for another completely different algorithm.

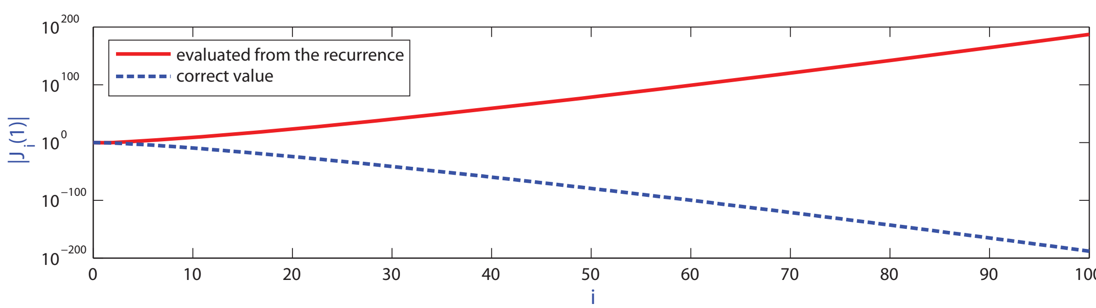

Figure 4: Behavior of the evaluated values of the Bessel function $J_i(1)$ using the recurrence (3).

This example shows that in many situations we have to take into account if the problem has a severe instability, that cannot be avoided using high-precision, or, on the contrary is unstable but it can be solved just by increasing the precision.

In some circumstances, the evaluation is stable, as in the evaluation of a Chebyshev series approximation of a function (except in "ill conditioned cases"), but in other cases we have a slightly unstable algorithm and we may have no other known option. In such cases a numerical analyst may work on finding a stable algorithm but an "applied user" needs a fast solution. So, one answer is the use of high-precision (in severe instabilities we have to search another method as commented before), as a fast option when no stable algorithm is known.

We next discuss two cases concerning with orthogonal polynomial series and the recurrences used in their evaluation. In the first example, the combined use of double-precision and high-precision controlled via theoretical bounds (running error bounds) permits to compute and evaluate series of Sobolev orthogonal polynomials (jus as an example of new families of orthogonal polynomials). And later, a modification, using ideas of high-precision computation, of the standard algorithm to evaluate Chebyshev series in ill-conditioned situations gives a relative accurate algorithm.

3.3.1 Sobolev orthogonal polynomials

The use of the classical families of orthogonal polynomials has been extended to almost all mathematical and physical disciplines, including approximation theory, spectral methods, representation of potentials and others. In the last few years, researchers have studied different extensions, like orthogonal polynomials in Sobolev spaces [51]. One particular case of interest is when measures related to derivatives are purely atomic, with a finite number of mass points. That is, given a set of K evaluation points $\{c_1, \ldots, c_K\}$ (the support of the discrete measure), a set of indexes that indicate the maximum order of derivatives in each evaluation point $\{r_1, \ldots, r_K\}$, and a set of non-negative coefficients $\{\lambda_{ji} \,|\, j = 1, \ldots, K;\, i = 0, \ldots, r_j\}$, we define the Sobolev inner product

$$\langle p, q \rangle_W = \int_{\mathbb{R}} p(x)\, q(x)\, \mathrm{d}\mu_0(x) + \sum_{j=1}^{K} \sum_{i=0}^{r_j} \lambda_{ji}\, p^{(i)}(c_j)\, q^{(i)}(c_j), \qquad \lambda_{ji} \geq 0. \tag{4}$$

This particular case is an important instance of the class of discrete Sobolev inner products. Thus, we want to study what happens when we face to the problem of evaluating a finite series of orthogonal polynomials with respect to this discrete Sobolev inner product. In [27, 30] it was proposed the algorithms for the generation and evaluation of these polynomials. The problem is that the algorithms are slightly unstable, and so, a combination of double and multiple precision is required. This may be done in such a way that the theoretical error bounds permit us to use high-precision just on the unstable cases, and so the computational complexity does not grows significatively.

In Figure 5 we show the behavior of some theoretical error bounds [30]: a backward error bound, the running error bound and the relative error in a multiple-precision evaluation of a Sobolev series. Note that we present relative error bounds and relative rounding errors, that is, for $q(x) \not\approx 0$ we divide by $|q(x)|$. We have up to degree 50 of the function $f(x) = (x+1)^2 \sin(4x)$ in Chebyshev-Sobolev orthogonal polynomials, considering one mass point $c = 1$ up to first derivative in the discrete part of the inner product. In the figures on the left we use double precision (53 bits on the mantissa) and on the right we use multiple precision (96 bits on the mantissa for $x < -0.5$ (on the left of the vertical line) and 64 for $x > -0.5$). The turning point

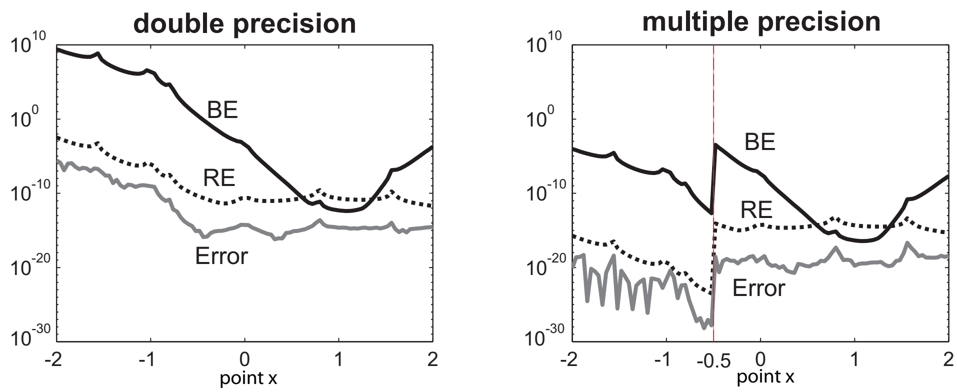

Figure 5: Behavior of the theoretical error bounds (BE a backward error bound and RE for the running error bound) and the relative error in the double- and multiple-precision evaluation of the Chebyshev-Sobolev approximation of degree 50 of the function $f(x) = (x+1)^2 \sin(4x)$, where the discrete Sobolev measure have one mass point $c = 1$ up to 1st derivative in the discrete part of the inner product. In the figure on the left we use double precision and on the right multiple-precision (on the left of the vertical line we use 96 bits on the mantissa and 64 on the right part). (Reproduced with permission from [30]).

$x = -0.5$ is the point where the relative running error in double precision is greater than 10^{-10}. Therefore, from the figures we can observe how the combined use of rounding error bounds (in this case the running error bound) and multiple-precision libraries permits us to evaluate Sobolev series accurately.

This example shows that although in some situations the recurrences to evaluate new families of orthogonal polynomials may lead to slightly unstable algorithms, the combination of theoretical bounds and high-precision techniques give rise to efficient and accurate algorithms.

3.3.2 Chebyshev series

Another situation where high precision is useful is in evaluating "ill-conditioned" polynomials. For instance, if one wishes to evaluate the polynomial $p(x) = (x-0.75)^7(x-1)^{10}$ close to one of its multiple roots, one will experience numerical difficulties. One solution is to find an optimal polynomial basis, although this may not be practical in many real-world situations. Another option is to use a good algorithm (e.g., Horner's algorithm for power series, the de-Calteljau's algorithm for the Bernstein basis and Clenshaw's algorithm for classical orthogonal polynomial basis), implemented with high-precision arithmetic. A third option, which is quite attractive when one does not want to deal with high-precision software, is to employ some ideas of compensated algorithms that recently emerged in stability analysis [79, 82]. This approach permits one to use double precision arithmetic, yet still maintain the quality of the numerical evaluations with a relative error on the order of the rounding unit u, plus the conditioning of the problem times the square of the rounding unit. The basis of these algorithms are the `TwoSum` and `TwoProd` schemes mentioned in Section 2. For instance, recently Graillat *et al.* [60] developed a "compensated" version of the Horner's algorithm. Also, H. Jiang *et al.* [70] developed a "compensated" version of Clenshaw's algorithm to evaluate a finite series of Chebyshev orthogonal polynomials $p(x) = \sum_{j=0}^{n} a_j T_j(x)$. For this compensated algorithm (and all the other ones) it is possible to prove the following relative error bounds:

Theorem 2 [70] *Let $p(x) = \sum_{i=0}^{n} a_i T_i(x)$ be a polynomial in Chebyshev form. If the condition number for polynomial evaluation of $p(x)$ at entry x is defined by*

$$\mathrm{cond}(p,x) = \frac{\widetilde{p}(|x|)}{|p(x)|} = \frac{\sum_{j=0}^{n} |a_j| \widetilde{T}_j(|x|)}{|\sum_{j=0}^{n} a_j T_j(x)|}, \tag{5}$$

with $\widetilde{T}_j(|x|)$ the absolute polynomials associated with $T_j(x)$ [70], then the relative forward error bounds of the Clenshaw algorithm *and* compensated Clenshaw algorithm *are such that*

$$\frac{|\mathtt{Clenshaw}(p,x) - p(x)|}{|p(x)|} \leq \mathcal{O}(u) \cdot \mathrm{cond}(p,x), \tag{6}$$

$$\frac{|\mathtt{CompClenshaw}(p,x) - p(x)|}{|p(x)|} \leq u + \mathcal{O}(u^2) \cdot \mathrm{cond}(p,x). \tag{7}$$

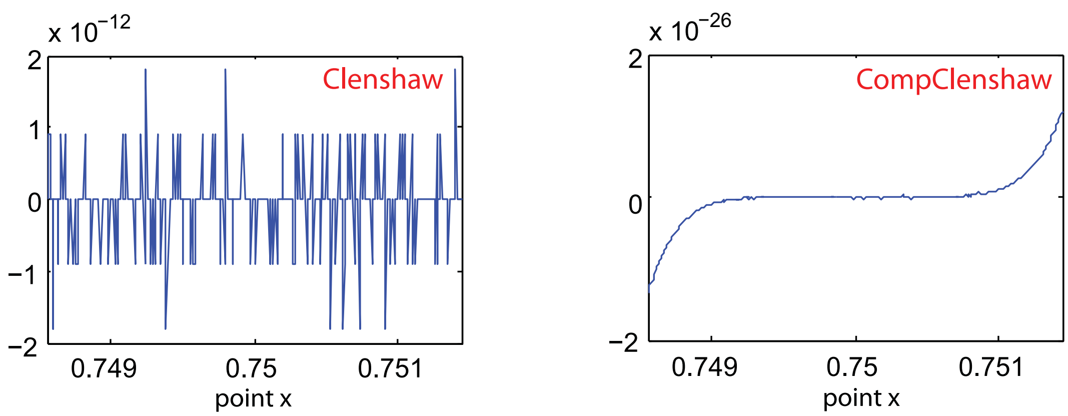

Figure 6: Evaluation of $p(x) = (x-0.75)^7(x-1)^{10}$ in the neighborhood of the multiple root $x = 0.75$, using the algorithms of Clenshaw (left) and Compensated Clenshaw (right). (Reproduced with permission from [70]).

This theorem shows one particularly nice feature of compensated algorithms, namely that the effect of the conditioning of the problem is delayed up to second order in the rounding unit u, yielding highly accurate (in relative error) computations.

Figure 6 presents the evaluation of the polynomial $p(x) = (x-0.75)^7(x-1)^{10}$ for 400 equally spaced points in the interval $[0.74855, 0.75145]$. It is clear that the compensated Clenshaw's algorithm gives a much smoother solution than the original Clenshaw's algorithm. Moreover, the relative error is always (except when $p(x)$ is very close to zero) of the order of the rounding unit u. This is often a crucial consideration in algorithms for locating zeros of polynomials in floating point arithmetic, because oscillations like the ones presented on the left figure can make impossible to obtain accurate results.

While compensated algorithms are often quite effective, they are not suitable for all situations, and so the use of high-precision software such as the QD library [69] is sometimes required.

3.4 Computing the "skeleton" of periodic orbits

In the words of Henri Poincaré, periodic orbits form the "skeleton" of a dynamical system and provide much useful information. Therefore, the search for periodic orbits is a quite old problem and numerous numerical and analytical methods have been designed for them. Here we mention just two methods that have been used with high-precision in the literature: the Lindstedt-Poincaré technique [89] and one of the most simple and powerful method to find periodic orbits, namely the systematic search method [24], where one takes advantage of symmetries of the system to find symmetric periodic orbits [74].

Theorem 3 *Let $o(\boldsymbol{x})$ be an orbit of a flow of an autonomous vector field $d\boldsymbol{x}/dt = f(\boldsymbol{x})$ with a reversal symmetry S (thus $dS(\boldsymbol{x})/dt = -f(S(\boldsymbol{x}))$). Then, an orbit $o(\boldsymbol{x})$ intersects* $\mathrm{Fix}(S) := \{\, \boldsymbol{x} \mid S(\boldsymbol{x}) = \boldsymbol{x} \,\}$ *in precisely two points if and only if the orbit is periodic (and not a fixed point) and symmetric with respect to S.*

The above results were already known by Birkhoff, DeVogelaere and Strömgren (among others) and were used to find symmetric periodic orbits.

The usage of high-precision numerical integrators in the determination of periodic orbits is required in the search of highly unstable periodic orbits. For instance, in Figure 7 we show the computed symmetric periodic orbit for the $7 + 2$ Ring problem using double and quadruple precision [26]. The $(n + 2)$-body Ring problem [26] describes the motion of an infinitesimal particle attracted by the gravitational field of $n+1$ primary bodies, n in the vertices of a regular polygon that is rotating on its own plane about the center with a constant angular velocity. Each point on the figures corresponds to the initial conditions of one symmetric periodic orbit, and the grey area corresponds to regions of forbidden motion (delimited by the limit curve). Note that in order to avoid "false" initial conditions it is useful to check if the initial conditions generate a periodic orbit up to a given tolerance level. But in the case of highly unstable periodic orbits we may lose several digits in each period, so that double precision is not enough in many unstable cases, resulting in gaps in the figure.

The Lindstedt-Poincaré method [89] for computing periodic orbits is based on the Lindstedt-Poincaré technique of perturbation theory, Newton's method for solving nonlinear systems and Fourier interpolation. D. Viswanath [90] uses this algorithm in combination with high-precision libraries to obtain periodic orbits for the Lorenz model at the classical Saltzman's parameter values. This procedure permits one to compute, to high accuracy (more than 100 digits of precision), highly unstable periodic orbits (for instance the orbit with symbolic dynamics $\mathrm{LRL}^2\mathrm{R}^2 \cdots \mathrm{L}^{15}\mathrm{R}^{15}$ has a leading characteristic multiplier 3.06×10^{59}, which means that we can expect that at each period we lose around 59 digits of precision). For these reasons, high-precision arithmetic plays a fundamental role in the study of the fractal properties of the Lorenz

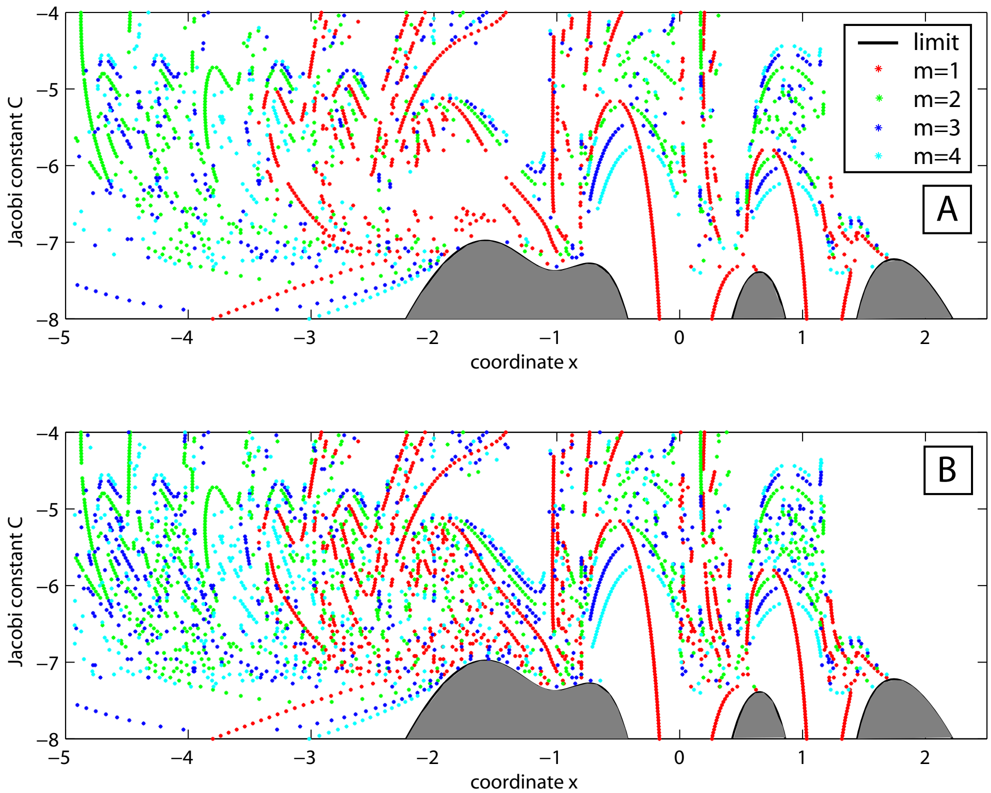

Figure 7: Symmetric periodic orbits (m denotes the multiplicity of the periodic orbit) in the most chaotic zone of the $7+2$ Ring problem using double (A) and quadruple (B) precision. (Reproduced with permission from [24]).

attractor (see Fig. 8) and in a consistent formal development of complex singularities of the Lorenz system using psi series [90, 91].

Another simpler option to compute high-precision periodic orbits has been proposed recently in [2], where the use of the Taylor series method permits to apply modified versions of the Newton method to obtain periodic orbits with more than 1000 precision digits. Figure 9 presents, as an example, the computational relative error vs. CPU time and number of iterations in the 1000 precision digits computation of the periodic orbits LR and LLRLR of the Lorenz model. The simulations have been done in a personal computer, what that means that nowadays any researcher is able to use such techniques in his/her research. This kind of studies permits to say that a deep and "microscopic" analysis in nonlinear dynamics is become available with the new algorithms and techniques of high-precision.

3.5 Divergent asymptotic series and homoclinic phenomena

One interesting phenomenon in dynamical systems arises in the study of the splitting of separatrices of area preserving maps. Numerical difficulties arise because this phenomena can exhibit

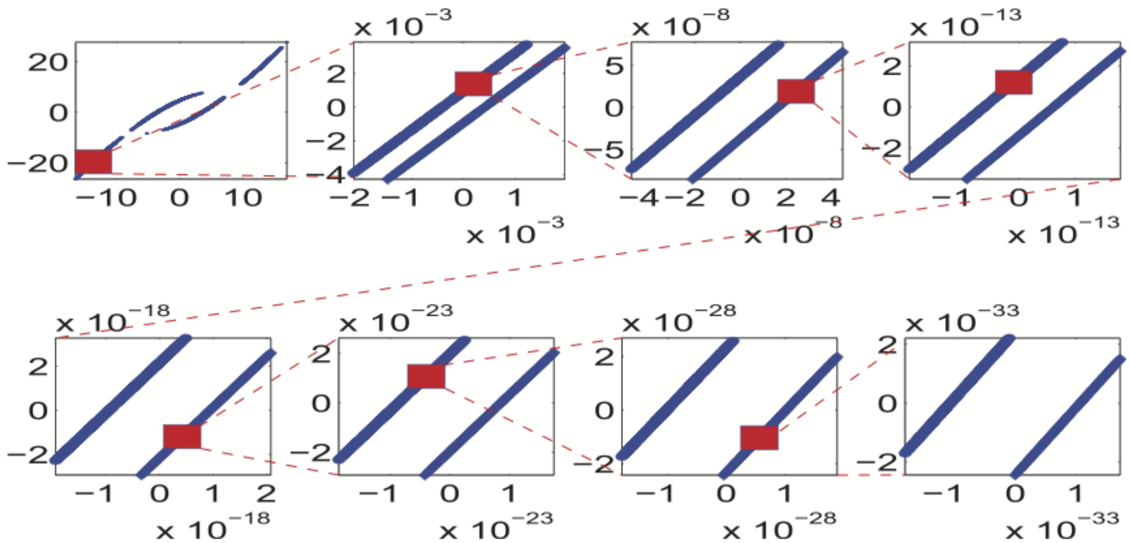

Figure 8: Fractal property of the Lorenz attractor. On the first plot, the intersection of an arbitrary trajectory on the Lorenz attractor with the section z = 27. The plot shows a rectangle in the $x-y$ plane. All later plots zoom in on a tiny region (too small to be seen by the unaided eye) at the center of the red rectangle of the preceding plot to show that what appears to be a line is in fact not a line. (Reproduced with permission from [91]).

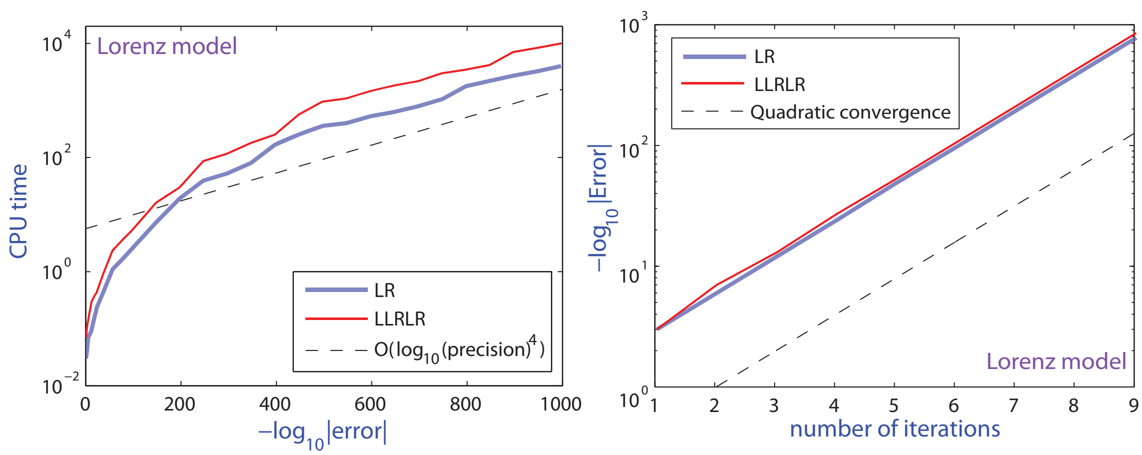

Figure 9: Computational relative error vs. CPU time and number of iterations in the 1000 precision digits computation of the periodic orbits LR and LLRLR of the Lorenz model. (Reproduced with permission from [2]).

exponentially small splitting [59, 64].

For instance, the most common paradigmatic example is the standard map defined by

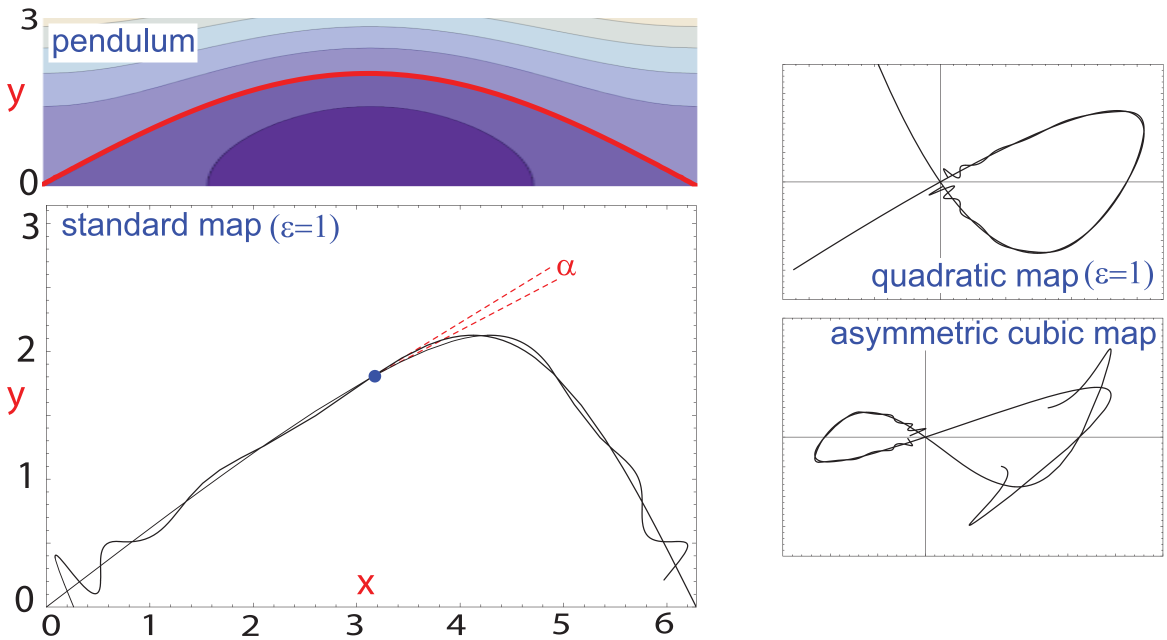

Figure 10: Left: Phase-space for the pendulum equations with the separatrix in red and the discrete version (standard map) for $\varepsilon = 1$ with the stable and the unstable separatrices. Right: stable and the unstable separatrices for the quadratic map and the asymmetric cubic map. (Partially reproduced with permission from [59])

$(x, y) \mapsto (\hat{x}, \hat{y})$ where

$$\hat{y} = y + \varepsilon \sin x, \quad \hat{x} = x + \hat{y}$$

and ε is a small positive constant. This map can be obtained, for example, by a simple time discretization (a symplectic Euler of discretization step $\sqrt{\varepsilon}$) of the pendulum equation $\dot{x} = y, \dot{y} = \sin x$ [64]. The phase space structure of both systems, the continuous case and the map, are very different (except for small values of ε). In fact, the pendulum problem is an integrable system and its phase space is very regular (see Fig. 10). There is a unique separatrix that connect the hyperbolic fixed point at 0 and at 2π, that is, the unstable manifold at 0 coincide with the stable manifold at 2π. When we see the map, the two manifolds do not coincide and so the separatrix splits (splitting of separatrices). Now we have transverse intersection points that gives homoclinic points and that imply the existence of complex dynamics or chaotic motion. Therefore the study of this phenomena of splitting of separatrices gives a deep information about the system, and so related with this, it is important to study the angle between the stable and the unstable separatrices at the intersection points. If the angle does not vanish we may affirm that this phenomena occurs. In Fig. 10 we illustrate also the phenomena with two other maps (the quadratic map and the asymmetric cubic map [59]).

An asymptotic formula for the angle between the stable and the unstable separatrices for the standard map at the primary homoclinic point was given by Lazutkin [76]:

$$\alpha = \frac{\pi}{\varepsilon} \mathrm{e}^{-\frac{\pi^2}{\sqrt{\varepsilon}}} \big(1118.8277059409\ldots + \mathcal{O}(\sqrt{\varepsilon})\big).$$

As a result, the separatrices are transversal, but the angle between them is exponentially small compared to ε. This leads to severe problems in numerical simulations. Gelfreich and Simó [59] use a homoclinic invariant ω that gives the area of a parallelogram defined by two vectors tangent to the stable and the unstable manifolds at the homoclinic point. While ω in the standard map can be represented by an asymptotic series, one question is what happens when we use several generalizations of the standard map. In [59], the authors employed high-precision computation of the homoclinic invariant and consecutive extraction of coefficients of an asymptotic expansion, in order to obtain a numerical evidence that various different types of asymptotic expansions arise in this class of problems. These results are unachievable using standard double precision; in some numerical simulations 1000-digit precision was required. In the literature there are other numerous examples of high-precision computation of this phenomena of exponentially small splitting of separatrices.

3.6 Detecting Strange Nonchaotic Attractors

In the study of dynamics of dissipative systems the detection of the attractors is quite important, because they are the visible invariant sets of the dynamics of the problem. An attractor is defined as *strange* if it is not a piecewise smooth manifold and *chaotic* if any orbit on it exhibits sensitive dependence on initial conditions. All the first examples of strange attractors in the literature where strange chaotic attractors, but soon some strange nonchaotic attractors (SNAs) were identified [62]. Several authors suggested that in the transition to chaos in quasiperiodically forced dissipative systems, in particular in the so called fractalization route in which a smooth torus seems to fractalize, strange nonchaotic attractors appear. In [65], Haro and Simó showed that in truth some of these attractors are nonstrange. These authors found that multiprecision arithmetic with more than 30 digits was needed to reliably study this behavior at very small scales. For example, in Fig. 11 we show the attractor of the RH map given by

$$\begin{aligned} x_{n+1} &= 1 + y_n - a x_n^2 + \varepsilon \cos(2\pi\theta_n), \\ y_{n+1} &= b x_n, \\ \theta_{n+1} &= \theta_n + \omega \qquad\qquad (\mathrm{mod}\ 1). \end{aligned}$$

This model is expected to be the scenario of the creation of SNAs through the fractalization route in which a smooth torus seems to fractalize, but although for low to double-precision simulations the attractor seems to be strange (see first, zoom-1 and zoom-2 pictures of Fig. 11), when one go to a scale 10^{-26} we may appreciate that the attractor do not seems to be strange (see zoom-3 picture of Fig. 11). Therefore, in this case (and in many cases) the SNAs is not produced via the fractalization route, but what is evident is that this phenomena requires a very high-precision numerical simulation to give a correct information of what really happens on the systems.

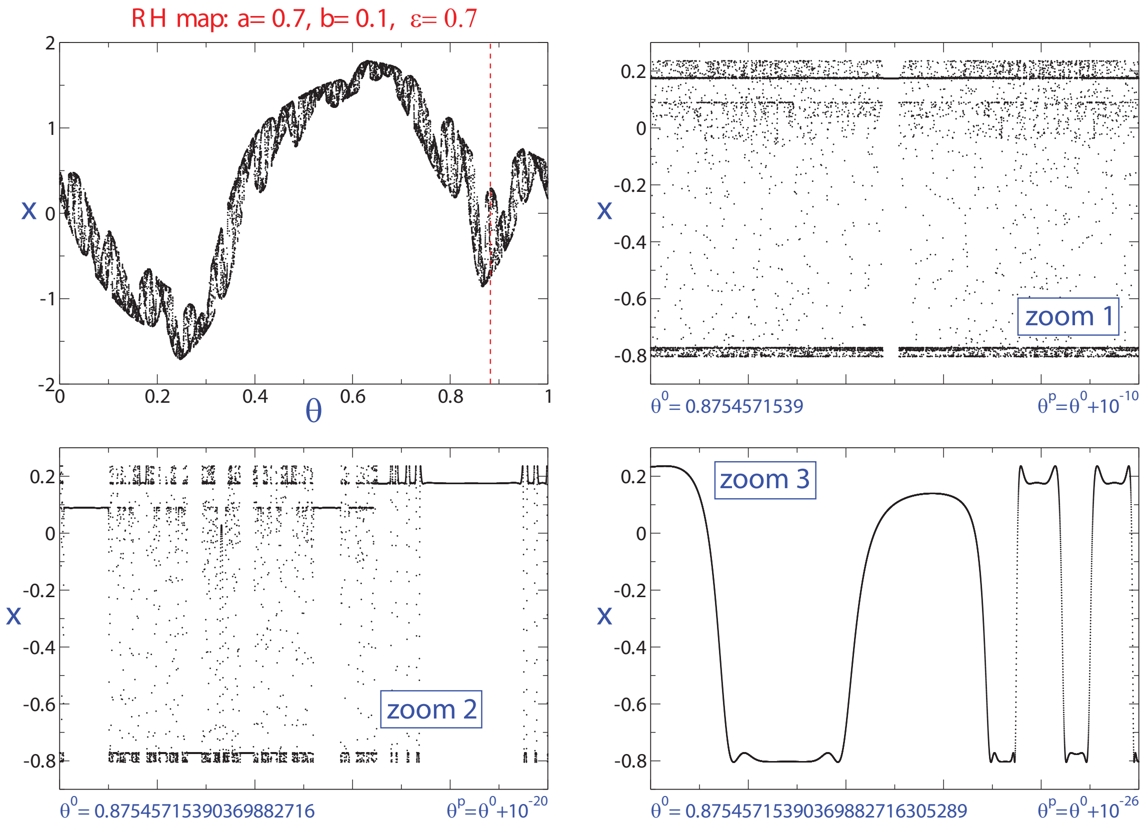

Figure 11: The attractor of the RH map with $a = 0.7$, $b = 0.1$, $\omega = (\sqrt{5} - 1)/2$ and $\varepsilon = 0.7$, and several zooms. (Reproduced with permission from [65]).

In some systems the Lyapunov sum can display arbitrarily large oscillations around the average line [45]. This means that, if the oscillations are wide enough, roundoff errors are locally amplified by a large factor. This may give us a numerically observed behavior which is completely wrong. An interesting example is the Logistic family driven by a rigid rotation [45] given by

$$\begin{aligned} x_{n+1} &= 1 - (a + \varepsilon \sin(2\pi\theta_n))x_n^2, \\ \theta_{n+1} &= \theta_n + \alpha \qquad \pmod 1). \end{aligned}$$

In this case we may observe how a numerically computed orbit can depend strongly on the precision used in its computation. Note that in this system the Lyapunov sum decreases during the first 600 iterates to the minimum value of -665, later increases in the next 800 iterates till -460 and decreases again, and so on (with an average line that decreases). This means that the local errors increase by about $\exp(-460 + 665) \approx 10^{89}$ in 1400 iterates. And as result we can imagine that using a precision lower than 10^{-89} will lead to erroneous results. In Fig. 12 we may observe the consequences, double precision and 60 digits lead to what it seems to be a SNAs. But if we repeat the calculus with 150 digits we observe that this was just an spurious result of insufficient accuracy on the simulations.

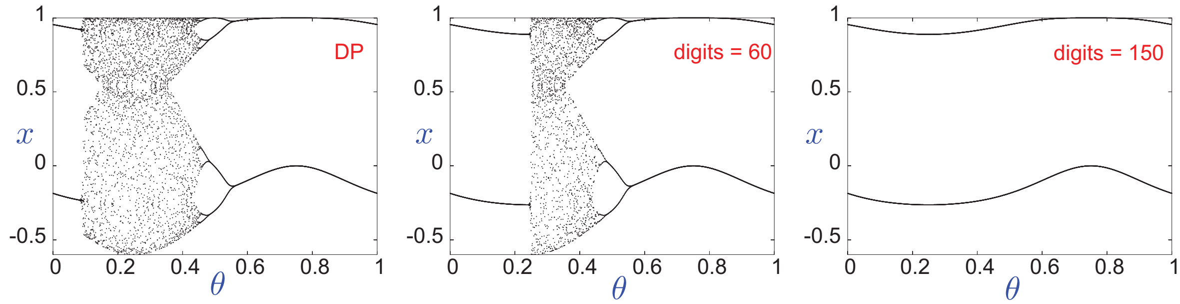

Figure 12: Attractors of the Logistic family driven by a rigid rotation with standard double precision (DP), 60 and 150 decimal digits, for $(a, \varepsilon, \alpha) = (1.30, 0.30, \gamma/1000)$, where γ denotes the Golden Mean. (Reproduced with permission from [45]).

4 Experimental mathematics

In this section we give a selection of five less directly applied applications:

4.1 Integer relation methods and digit formulas

Very high-precision computations (typically 100 to several thousand digits) have proven to be an essential tool for the emerging discipline of "experimental mathematics" [35, 8]. One of the key techniques used here is the PSLQ integer relation detection algorithm [15], which, given an n-long vector (x_i) of real numbers (presented as a vector of high-precision values), attempts to recover the integer coefficients (a_i), not all zero, such that

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = 0 \tag{8}$$

(to available precision), or else determines that there are no such integers (a_i) of a given size. The PSLQ algorithm operates by developing, iteration by iteration, an integer-valued matrix A that successively reduces the maximum absolute value of the entries of the vector $y = Ax$ (where x is the input vector mentioned above), until one of the entries of y is zero or within an "epsilon" of zero. With PSLQ or any other integer relation detection scheme, if the underlying integer relation vector of length n has entries of maximum size d digits, then the input data must be specified to at least nd-digit precision (and the algorithm must be performed using this precision level) or else the true relation will be lost in a sea of spurious artifacts of numerical round-off error.

Perhaps the best-known application of PSLQ in experimental mathematics is the 1996 computer-based discovery of what is now known as the "BBP" formula for π:

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right). \tag{9}$$

This formula has the remarkable property that it permits one to calculate binary or hexadecimal digits beginning at the n-th digit, without needing to calculate any of the first $n-1$ digits, using a simple scheme that requires very little memory and no multiple-precision arithmetic software [7], [35, pp. 135–143]. In 2010, Tse Wo Zse, a researcher with Yahoo! Cloud Computing, used a variant of this formula to compute a string of hexadecimal digits of π beginning at the 500 trillionth digit (corresponding to the two quadrillionth binary digit) [88].

Since 1996, numerous other formulas of this type have been found using PSLQ and then subsequently proven [6]. Here are three example — a base 2^6 formula for π^2, a base 3^6 formula for π^2 and a base 2^{12} formula for Catalan's constant $G = \sum_{n=0}^{\infty} (-1)^n/(2n+1)^2 = 0.91596559417722\ldots$:

$$\pi^2 = \frac{9}{8} \sum_{k=0}^{\infty} \frac{1}{64^k} \left(\frac{16}{(6k+1)^2} - \frac{24}{(6k+2)^2} - \frac{8}{(6k+3)^2} - \frac{6}{(6k+4)^2} + \frac{1}{(6k+5)^2} \right). \tag{10}$$

$$\begin{aligned} \pi^2 = {} & \frac{2}{27} \sum_{k=0}^{\infty} \frac{1}{729^k} \left(\frac{243}{(12k+1)^2} - \frac{405}{(12k+2)^2} - \frac{81}{(12k+4)^2} - \frac{27}{(12k+5)^2} \right. \\ & \left. - \frac{72}{(12k+6)^2} - \frac{9}{(12k+7)^2} - \frac{9}{(12k+8)^2} - \frac{5}{(12k+10)^2} + \frac{1}{(12k+11)^2} \right). \end{aligned} \tag{11}$$

$$\begin{aligned} G = {} & \frac{1}{4096} \sum_{k=0}^{\infty} \frac{1}{4096^k} \left(\frac{36864}{(24k+2)^2} - \frac{30720}{(24k+3)^2} - \frac{30720}{(24k+4)^2} - \frac{6144}{(24k+6)^2} - \frac{1536}{(24k+7)^2} \right. \\ & + \frac{2304}{(24k+9)^2} + \frac{2304}{(24k+10)^2} + \frac{768}{(24k+14)^2} + \frac{480}{(24k+15)^2} + \frac{384}{(24k+11)^2} + \frac{1536}{(24k+12)^2} \\ & \left. + \frac{24}{(24k+19)^2} - \frac{120}{(24k+20)^2} - \frac{36}{(24k+21)^2} + \frac{48}{(24k+22)^2} - \frac{6}{(24k+23)^2} \right). \end{aligned} \tag{12}$$

As with the original BBP formula for π, these formulas permit one to directly calculate digits beginning at an arbitrary starting point, thus making accessible mathematical objects that until very recently had widely been regarded as forever inaccessible to human reasoning or machine computation. In 2011, equation (10) was employed to calculate base-64 digits of π^2; equation (11) was employed to calculate base-729 digits of π^2; and equation (12) was employed to calculate base-4096 digits of G; in each case beginning at the ten trillionth position and validated by a second independent computation. The resulting base-8 digit string of π^2 is

$$601145053032364757245000057432627545303630524163506 34$$

(with each pair of base-8 digits corresponding to a base-64 digit). The resulting base-9 digit string of π^2 is

$$122644850645485831771111352101628560483234534 68$$

(with each triplet of base-9 digits corresponds to one base-729 digit). The resulting base-8 string of Catalan's constant G is

$$347050537747770511226133716201252573272173245 22$$

(with each quadruplet of base-8 digits corresponding to one base-4096 digit). This suite of computations involved 1.549×10^{19} floating point operations, which is comparable to the cost of the most sophisticated animated movies as of the present time (2011). However, a clever choice of formulas, again discovered using a high-precision PSLQ program, resulted in significant savings particularly for the Catalan constant calculation [14].

4.1.1 BBP series and normality

In an unexpected turn of events, it has been found that these computer-discovered formulas have implications for the age-old question of whether (and why) the digits of certain well-known math constants are statistically random. In particular, one of the present researchers and Richard Crandall found that the question of whether constants such as π and $\log 2$ are 2-normal (i.e., every string of m binary digits appears, in the limit, with frequency 2^{-m}) reduces to a conjecture about the behavior of a certain explicit pseudorandom number generator that is related to the respective BBP-type formula for that constant [16], [35, pp. 163–178]. This same line of investigation has led to a formal proof of normality for an uncountably infinite class of explicit real numbers [17], the simplest instance of which is

$$\alpha_{2,3} \quad = \quad \sum_{n=1}^{\infty} \frac{1}{3^n 2^{3^n}},$$

which is provably 2-normal.

4.2 Nonlinear oscillator theory

One application of experimental mathematical techniques to a mathematical physics problem was inspired by a recent paper by Quinn, Rand, and Strogatz. They described a nonlinear coupled oscillator system in which phase transition was described by means of the formula

$$0 = \sum_{i=1}^{N} \left(2\sqrt{1 - s^2(1 - 2(i-1)/(N-1))^2} - \frac{1}{\sqrt{1 - s^2(1 - 2(i-1)/(N-1))^2}} \right). \tag{13}$$

They noted that for large N, $s \approx 1 - c/N$, where c = 0.6054436... . These researchers asked two of the present authors and Richard Crandall to validate and extend this computation, and challenged us to identify this limit if it exists. By means of a Richardson extrapolation scheme, implemented on 64-CPUs of a highly parallel computer system, we computed (using the QD software)

$$c = 0.60544365719673274947892284244720747522089960\ldots$$

This calculation led to a *proof* that the limit c exists and is the positive root of the *Hurwitz zeta* function

$$\zeta\left(1/2, c/2\right) = 0,$$

where $\zeta(s,a) := \sum_{n\geq 0} 1/(n+a)^s$. Moreover, we were able to sketch the higher-order asymptotic behavior [12], something that would have been impossible without discovery of an analytic formula.

Such systems are especially interesting in light of even more recent work by Steve Strogatz and his collaborators on *chimera* — coupled systems which can self-organize in parts of their domain and remain disorganized elsewhere. See Figure 13, taken from [4], in which (b) in the left panel shows a coupled system which has partial self-organized and (c) shows how well prediction and simulation agree.

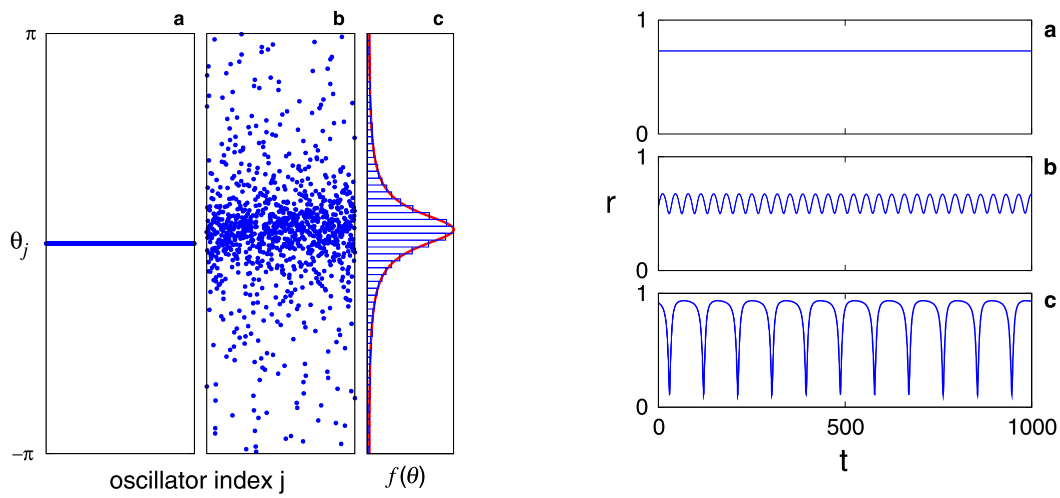

Figure 13: Simulated chimera. (Left) Snapshot of a chimera state, obtained by numerical integration. (a) Synchronized population. (b) Desynchronized population. (c) Predicted density of desynchronized phases (smooth curve) agrees with observed histogram. (Right) Order parameter r versus time. (a) stable chimera; (b) breathing chimera; (c) long-period breather. Numerical integration began from an initial condition close to the chimera state, and plots shown begin after allowing a transient time of 2000 units. (Figures and parameters from [4])

4.3 Ising integrals

Very high-precision computations, combined with the PSLQ algorithm, have been remarkably effective in recognizing (in terms of analytic formulas) certain classes of definite integrals that arise in mathematical physics settings. Such results are highly prized by mathematical physicists, because they can be used in asymptotic expansions or other useful analytic expressions. Results of this sort remain hidden if one merely computes standard-precision numerical values.

These studies most often have employed either Gaussian quadrature (in cases where the function is well behaved in a closed interval) or the "tanh-sinh" quadrature scheme due to Takahasi and Mori [87] (in cases where the function has an infinite derivative or blow-up singularity at one or both endpoints). For many integrand functions, these schemes exhibit "quadratic" or "exponential" convergence — dividing the integration interval in half (or, equivalently, doubling the number of evaluation points) approximately doubles the number of correct digits in the result [19].

In one study, the tanh-sinh quadrature scheme, implemented using the ARPREC software, was employed to study the following classes of integrals [11]. Here, the D_n integrals arise in the Ising theory of mathematical physics, and the C_n have tight connections to quantum field theory:

$$
\begin{aligned}
C_n &= \frac{4}{n!}\int_0^\infty \cdots \int_0^\infty \frac{1}{\left(\sum_{j=1}^n (u_j + 1/u_j)\right)^2} \frac{\mathrm{d}u_1}{u_1} \cdots \frac{\mathrm{d}u_n}{u_n} \\
D_n &= \frac{4}{n!}\int_0^\infty \cdots \int_0^\infty \frac{\prod_{i<j} \left(\frac{u_i - u_j}{u_i + u_j}\right)^2}{\left(\sum_{j=1}^n (u_j + 1/u_j)\right)^2} \frac{\mathrm{d}u_1}{u_1} \cdots \frac{\mathrm{d}u_n}{u_n} \\
E_n &= 2\int_0^1 \cdots \int_0^1 \left(\prod_{1 \le j < k \le n} \frac{u_k - u_j}{u_k + u_j}\right)^2 \mathrm{d}t_2\, \mathrm{d}t_3 \cdots dt_n,
\end{aligned}
$$

where (in the last line) $u_k = \prod_{i=1}^k t_i$.

Needless to say, evaluating these n-dimensional integrals to high precision presents a daunting computational challenge. Fortunately, in the first case, we were able to show that the C_n integrals can be written as one-dimensional integrals:

$$
C_n = \frac{2^n}{n!}\int_0^\infty p K_0^n(p)\, \mathrm{d}p,
$$

where K_0 is the *modified Bessel function* [3]. After computing C_n to 1000-digit accuracy for various n, we were able to identify the first few instances of C_n in terms of well-known constants, e.g., $C_4 = 7\zeta(3)/12$, where ζ denotes the Riemann zeta function. When we computed C_n for fairly large n, for instance

$$
C_{1024} = 0.63047350337438679612204019271087890435458707871273234\ldots,
$$

we found that these values rather quickly approached a limit. By using the new edition of the *Inverse Symbolic Calculator*, available at `http://carma-lx1.newcastle.edu.au:8087`, this numerical value can be identified as

$$
\lim_{n \to \infty} C_n = 2e^{-2\gamma},
$$

where γ is Euler's constant, which we were subsequently able to prove [11].

The integrals D_n and E_n are much more difficult to evaluate, since they are not reducible to one-dimensional integrals (as far as we can tell), but with certain symmetry transformations and symbolic integration we were able to reduce the dimension in each case by one or two. In the case of D_5 and E_5, the resulting 3-D integrals are extremely complicated, but we were nonetheless able to numerically evaluate these to at least 240-digit precision using highly parallel computer systems at Virginia Tech and at the Lawrence Berkeley National Lab. In this way, we produced the following evaluations, all of which except the last we subsequently were able to prove:

$$\begin{aligned}
D_2 &= 1/3 \\
D_3 &= 8 + 4\pi^2/3 - 27\,\mathrm{L}_{-3}(2) \\
D_4 &= 4\pi^2/9 - 1/6 - 7\zeta(3)/2 \\
E_2 &= 6 - 8\log 2 \\
E_3 &= 10 - 2\pi^2 - 8\log 2 + 32\log^2 2 \\
E_4 &= 22 - 82\zeta(3) - 24\log 2 + 176\log^2 2 - 256(\log^3 2)/3 + 16\pi^2\log 2 - 22\pi^2/3 \\
E_5 &\stackrel{?}{=} 42 - 1984\,\mathrm{Li}_4(1/2) + 189\pi^4/10 - 74\zeta(3) - 1272\zeta(3)\log 2 + 40\pi^2\log^2 2 \\
&\quad -62\pi^2/3 + 40(\pi^2\log 2)/3 + 88\log^4 2 + 464\log^2 2 - 40\log 2,
\end{aligned}$$

where Li denotes the polylogarithm function. In the case of D_2, D_3 and D_4, these are confirmations of known results. We tried but failed to recognize D_5 in terms of similar constants (the 500-digit numerical value is available if anyone wishes to try). The conjectured identity shown here for E_5 was confirmed to 240-digit accuracy, which is 180 digits beyond the level that could reasonably be ascribed to numerical round-off error; thus we are quite confident in this result even though we do not have a formal proof [11].

4.4 Random walks

A more recent numerical study considered, for complex s, the n-dimensional *ramble integrals* [9]

$$W_n(s) \;=\; \int_{[0,1]^n} \left| \sum_{k=1}^{n} e^{2\pi x_k i} \right|^s \mathrm{d}x, \tag{14}$$

which occur in the theory of uniform random walk integrals in the plane, where at each step a unit-step is taken in a random direction. Integrals such as (14) are the s-th moment of the distance to the origin after n steps. It is shown in [42] that when $s = 0$ the first derivatives of these integrals can be written as

$$W_n'(0) \;=\; \log(2) - \gamma - \int_0^1 \left(J_0^n(x) - 1\right)\frac{\mathrm{d}x}{x} - \int_1^\infty J_0^n(x)\frac{\mathrm{d}x}{x} \tag{15}$$

$$\;=\; \log(2) - \gamma - n\int_0^\infty \log(x) J_0^{n-1}(x) J_1(x)\mathrm{d}x, \tag{16}$$

where $J_n(x)$ denotes the Bessel function of the first kind.

Due to the oscillatory nature of these integrals, they present substantial challenges for high-precision numerical integration. One approach that we have found effective for these integrals is known as the Sidi mW extrapolation algorithm, as described in a 1994 paper by Lucas and Stone [75] (which in turn is based on two earlier papers by Sidi [83, 84]), combined with tanh-sinh quadrature and Gaussian quadrature [9]. Using this scheme, we were able to evaluate these integrals to 1000-digit accuracy, at least when n is odd, using the ARPREC software. This scheme is not very effective when n is even, but in this case we were able to compute modestly high precision results (50–100 digits) by employing asymptotic formulas for the Bessel function. In response to this ineffectiveness, Sidi [85] has made an analysis and proposed a more sophisticated scheme which should redress the situation.

These results were used to verify several other studies. For instance, our result when $n = 6$ matched to 80-digit precision a computation based on a conjecture due to Villegas [43]. Similarly, for $n = 4$ our 80-digit result agrees to full precision with the closed form given in [42].

Our calculations also confirmed, to 600-digit precision, the following amazing conjecture based on one of Villegas, [43]:

$$W_5'(0) \stackrel{?}{=} \left(\frac{15}{4\pi^2}\right)^{5/2} \int_0^\infty \left\{\eta^3(e^{-3t})\eta^3(e^{-5t}) + \eta^3(e^{-t})\eta^3(e^{-15t})\right\} t^3 \, \mathrm{d}t, \tag{17}$$

where

$$\eta(q) = q^{1/24} \prod_{n\geq 1}(1-q^n) = q^{1/24} \sum_{n=-\infty}^{\infty} (-1)^n q^{n(3n+1)/2}. \tag{18}$$

While the intuitive genesis of equation (17) lies in algebraic K-theory, it is fair to say that there is no inkling of how to prove it.

4.5 Moments of elliptic products

The research on ramble integrals also led us to examine moments of elliptic integral functions of the form [9]:

$$I(n_0, n_1, n_2, n_3, n_4) = \int_0^1 x^{n_0} K^{n_1}(x) K'^{n_2}(x) E^{n_3}(x) E'^{n_4}(x) \mathrm{d}x, \tag{19}$$

where the elliptic functions K, E and their complementary versions are given by:

$$\begin{aligned} K(x) &= \int_0^1 \frac{\mathrm{d}t}{\sqrt{(1-t^2)(1-x^2t^2)}} \qquad K'(x) = K(\sqrt{1-x^2}) \\ E(x) &= \int_0^1 \frac{\sqrt{1-x^2t^2}}{\sqrt{1-t^2}} \, \mathrm{d}t \qquad E'(x) = E(\sqrt{1-x^2}). \end{aligned} \tag{20}$$

To better understand these product integrals, we computed a large number of them (4389 individual integrals in total) to extreme precision — 1500 to 3000-digit precision — using the ARPREC software. We then discovered, using PSLQ, thousands of intriguing relations between these numerical values, including the following limited selection [9]:

$$\begin{aligned}81\int_0^1 x^3K^2(x)E(x)\mathrm{d}x &\stackrel{?}{=} -6\int_0^1 K^3(x)\mathrm{d}x - 24\int_0^1 x^2K^3(x)\mathrm{d}x\\&+51\int_0^1 x^3K^3(x)\mathrm{d}x + 32\int_0^1 x^4K^3(x)\mathrm{d}x \qquad (21)\\-243\int_0^1 x^3K(x)E(x)K'(x)\mathrm{d}x &\stackrel{?}{=} -59\int_0^1 K^3(x)\mathrm{d}x + 468\int_0^1 x^2K^3(x)\mathrm{d}x\\&+156\int_0^1 x^3K^3(x)\mathrm{d}x - 624\int_0^1 x^4K^3(x)\mathrm{d}x - 135\int_0^1 xK(x)E(x)K'(x)\mathrm{d}x \qquad (22)\\-20736\int_0^1 x^4E^2(x)K'(x)\mathrm{d}x &\stackrel{?}{=} 3901\int_0^1 K^3(x)\mathrm{d}x - 3852\int_0^1 x^2K^3(x)\mathrm{d}x\\&-1284\int_0^1 x^3K^3(x)\mathrm{d}x + 5136\int x^4K^3(x)\mathrm{d}x - 2592\int_0^1 x^2K^2(x)K'(x)\mathrm{d}x\\&-972\int_0^1 K(x)E(x)K'(x)\mathrm{d}x - 8316\int_0^1 xK(x)E(x)K'(x)\mathrm{d}x. \qquad (23)\end{aligned}$$

These identities led to a detailed study by James Wan [92] who has been able to prove many but by no means all of them.

5 Other brief examples

We briefly summarize here a number of other applications of high-precision arithmetic that have been reported to us. For additional details, please see the listed references.

5.1 Supernova simulations

Recently Edward Baron, Peter Hauschildt, and Peter Nugent used the QD package to solve for the non-local thermodynamic equilibrium populations of iron and other atoms in the atmospheres of supernovae and other astrophysical objects [20, 66]. Iron, for example, may exist as Fe II in the outer parts of the atmosphere, but in the inner parts Fe IV or Fe V could be dominant. Introducing artificial cutoffs leads to numerical glitches, so it is necessary to solve for all of these populations simultaneously. Since the relative population of any state from the dominant stage is proportional to the exponential of the ionization energy, the dynamic range of these numerical values can be large. Among various potential solutions, these authors found that using double-double (or, in some cases, quad-double) arithmetic to be the most straightforward and effective.

5.2 Climate modeling

It is well-known that climate simulations are fundamentally chaotic — if microscopic changes are made to the present state, within a certain period of simulated time the future state is completely different. Indeed, ensembles of these calculations are required to obtain statistical confidence in global climate trends produced from such calculations. As a result, climate modeling codes quickly diverge from any "baseline" calculation, even if only the number of processors used to run the code is changed. For this reason, it is often difficult for researchers to compare results, or even to determine whether they have correctly deployed their code on a given system. Recently Helen He and Chris Ding found that almost all of the numerical variation in an atmospheric code occurred in a long inner product loop in the data assimilation step and in a similar operation in a large conjugate gradient calculation. He and Ding found that employing double-double arithmetic for these loops dramatically reduced the numerical variability of the entire application, permitting computer runs to be compared for much longer run times than before [68].

5.3 Coulomb n-body atomic system simulations

Numerous computations have been performed using high-precision arithmetic to study atomic-level Coulomb systems. For example, Alexei Frolov of Queen's University in Ontario, Canada has used high-precision software to solve the generalized eigenvalue problem $(\hat{H} - E\hat{S})C = 0$, where the matrices $\hat{H}$ and $\hat{S}$ are large (typically $5,000 \times 5,000$ in size) and very nearly degenerate. Until recently, progress in this arena was severely hampered by the numerical difficulties induced by these nearly degenerate matrices. Frolov found that by employing 120-digit arithmetic, "we can consider and solve the bound state few-body problems which have been beyond our imagination even four years ago" [18, 57].

5.4 Studies of the fine structure constant of physics

In the past few years, significant progress has been achieved in using high-precision arithmetic to obtain highly accurate solutions to the Schrodinger equation for the lithium atom. In particular, the non-relativistic ground state energy has been calculated to an accuracy of a few parts in a trillion, a factor of 1500 improvement over the best previous results. With these highly accurate wave functions, researchers have been able to test the relativistic and QED effects at the 50 parts per million (ppm) level and also at the one ppm level [93]. Along this line, a number of properties of lithium and lithium-like ions have also been calculated, including the oscillator strengths for certain resonant transitions, isotope shifts in some states, dispersion coefficients and Casimir-Polder effects between two lithium atoms. When some additional computations are completed, the fine structure constant may be obtained to an accuracy of 16 parts per billion [94].

5.5 Scattering amplitudes of quarks, gluons and bosons

An international team of physicists working on the Large Hadron Collider (LHC) is computing scattering amplitudes involving quarks, gluons and gauge vector bosons, in order to predict what results could be expected on the LHC. By default, these computations are performed using conventional double precision (64-bit IEEE) arithmetic. Then if a particular phase space point is deemed numerically unstable, it is recomputed with double-double precision. These researchers expect that further optimization of the procedure for identifying unstable points may be required to arrive at an optimal compromise between numerical accuracy and performance. Their objective is to design a procedure where the number of digits in the higher precision calculation is dynamically set according to the instability of the point [54]. Three related applications of high-precision arithmetic are given in [34, 80, 48].

6 Conclusion

We have presented here a brief survey of the rapidly expanding applications of high-precision arithmetic in modern scientific computing. It is worth noting that all of these examples have arisen in the past ten years. Thus we may be witnessing the birth of a new era of scientific computing, in which the numerical precision required for a computation is as important to the program design as are the algorithms and data structures. We hope that our survey and analysis of these computations will be useful in this process.

References

[1] A. Abad, R. Barrio, F. Blesa and M. Rodriguez, "TIDES: a Taylor series Integrator for Differential EquationS," 2011, `http:gme.unizar.es/software/tides`.

[2] Alberto Abad, Roberto Barrio, and Angeles Dena, "Computing periodic orbits with arbitrary precision," *Phys. Rev. E*, vol. 84 (2011), 016701.

[3] M. Abramowitz and I. A. Stegun, ed., *Handbook of Mathematical Functions*, Dover, New York, 1972.

[4] D. M. Abrams, R. Mirollo, S. H. Strogatz and D. A. Wiley, "Solvable model for chimera states of coupled oscillators," *Physical Review Letters*, vol. 101 (2008), 084103.

[5] J. Applegate, M. Douglas, Y. Gursel, G. J. Sussman and J. Wisdom, "The outer solar system for 200 Million years," *Astronomical Journal*, vol. 92 (1986), 176–194.

[6] D. H. Bailey, "A compendium of BBP-type formulas," Apr. 2011, available at `http://crd.lbl.gov/~dhbailey/dhbpapers/bbp-formulas.pdf`. An interactive database is online at `http://bbp.carma.newcastle.edu.au`.

[7] D. H. Bailey, P. B. Borwein, and S. Plouffe, "On the rapid computation of various polylogarithmic constants," *Math. of Computation*, vol. 66 (Apr 1997), 903–913.

[8] D. H. Bailey and J. M. Borwein, "Experimental mathematics: Examples, methods and implications," *Notices of the AMS*, vol. 52 (May 2005), 502-514.

[9] D. H. Bailey and J. M. Borwein, "Hand-to-hand combat with thousand-digit integrals," *Journal of Computational Science*, to appear, `http://crd.lbl.gov/~dhbailey/dhbpapers/combat.pdf`.

[10] D. H. Bailey, J. M. Borwein, D. Broadhurst and M. L. Glasser, "Elliptic integral evaluations of Bessel moments," *J. Physics A: Math. and Gen.*, vol. 41 (2008), 205203.

[11] D. H. Bailey, J. M. Borwein and R. E. Crandall, "Integrals of the Ising class," *J. Physics A: Math. and Gen.*, vol. 39 (2006), 12271–12302.

[12] D. H. Bailey, J. M. Borwein and R. E. Crandall, "Resolution of the Quinn-Rand-Strogatz constant of nonlinear physics," *Exp. Mathematics*, vol. 18 (2009), 107–116.

[13] D. H. Bailey, D. Borwein, J. M. Borwein and R. Crandall, "Hypergeometric forms for Ising-class integrals," *Exp. Mathematics*, vol. 16 (2007), 257–276.

[14] D. H. Bailey, J. M. Borwein, A. Mattingly and G. Wightwick, "The computation of previously inaccessible digits of π^2 and Catalans constant," *Notices of the AMS*, to appear, 2011, `http://crd.lbl.gov/~dhbailey/dhbpapers/bbp-bluegene.pdf`.

[15] D. H. Bailey and D. Broadhurst, "Parallel integer relation detection: Techniques and applications," *Math. of Computation*, vol. 70 (2000), 1719–1736.

[16] D. H. Bailey and R. E. Crandall, "On the random character of fundamental constant expansions," *Exp. Mathematics*, vol. 10 (2001), 175–190.

[17] D. H. Bailey and R. E. Crandall, "Random generators and normal numbers," *Exp. Mathematics*, vol. 11 (2004), 527–546.

[18] D. H. Bailey and A. M. Frolov, "Universal variational expansion for high-precision bound-state calculations in three-body systems. Applications to weakly-bound, adiabatic and two-shell cluster systems," *J. Physics B*, vol. 35 (2002), 42870–4298.

[19] D. H. Bailey, X. S. Li and K. Jeyabalan, "A comparison of three high-precision quadrature schemes," *Exp. Mathematics*, vol. 14 (2005), 317–329.

[20] E. Baron and P. Nugent, personal communication, Nov. 2004.

[21] R. Barrio, "Rounding error bounds for the Clenshaw and Forsythe algorithms for the evaluation of orthogonal polynomial series," *J. Comput. Appl. Math.* 138 (2002) 1985–204.

[22] R. Barrio, "Performance of the Taylor series method for ODEs/DAEs," *Appl. Math. Comput.*, vol. 163 (2005), 525–545.

[23] R. Barrio, "Sensitivity analysis of ODEs/DAEs using the Taylor series method," *SIAM Journal on Scientific Computing*, vol. 27 (2006), 1929–1947.

[24] R. Barrio and F. Blesa, "Systematic search of symmetric periodic orbits in 2DOF Hamiltonian systems," *Chaos, Solitons and Fractals*, vol. 41 (2009), 560–582.

[25] R. Barrio, F. Blesa, M. Lara, "VSVO formulation of the Taylor method for the numerical solution of ODEs," *Comput. Math. Appl.*, vol. 50 (2005), 93–111.

[26] R. Barrio, F. Blesa and S. Serrano, “Qualitative analysis of the $(n+1)$-body ring problem,” *Chaos Solitons Fractals*, vol. 36 (2008), 1067–1088.

[27] R. Barrio, B. Melendo and S. Serrano, “Generation and evaluation of orthogonal polynomials in discrete Sobolev spaces I. Algorithms,” *J. Comput. Appl. Math.*, vol. 181 (2005), 280–298.

[28] R. Barrio, B. Melendo and S. Serrano, “On the numerical evaluation of linear recurrences,” *Journal of Computational and Applied Mathematics*, vol. 150 (2003), 71–86.

[29] R. Barrio, M. Rodríguez, A. Abad, and F. Blesa, “Breaking the limits: the Taylor series method,” *Applied Mathematics and Computation*, vol. 217 (2011), 7940–7954

[30] R. Barrio and S. Serrano, “Generation and evaluation of orthogonal polynomials in discrete Sobolev spaces II. Numerical stability,” *J. Comput. Appl. Math.*, vol. 181 (2005), 299–320.

[31] H. H. Bauschke, P. L. Combettes, and D. R. Luke, “Finding best approximation pairs relative to two closed convex sets in Hilbert spaces,” *J. Approx. Theory*, vol. 127 (2004), 178–192.

[32] H. H. Bauschke, P. L. Combettes, and D. R. Luke, “Phase retrieval, error reduction algorithm, and Fienup variants: A view from convex optimization,” *J. Opt. Soc. Amer. A*, vol. 19 (2002), 1334–1345.

[33] H. H. Bauschke, P. L. Combettes, and D. R. Luke, “A strongly convergent reflection method for finding the projection onto the intersection of two closed convex sets in a Hilbert space,” *J. Approx. Theory*, vol. 141 (2006), 63–69.

[34] C. F. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, H. Ita, D. A. Kosower and D. Maitre, “An automated implementation of on-shell methods for one-loop amplitudes,” *Phys. Rev. D*, vol. 78 (2008), 036003, `http://arxiv.org/abs/0803.4180`.

[35] J. M. Borwein and D. H. Bailey, *Mathematics by Experiment: Plausible Reasoning in the 21st Century*, A.K. Peters, Natick, MA, second edition, 2008.

[36] J. M. Borwein and D. H. Bailey, *Experimentation in Mathematics: Computational Paths to Discovery*, A.K. Peters, Natick, MA, 2004.

[37] J. M. Borwein and P. B. Borwein, “The arithmetic-geometric mean and the fast computation of elementary functions,” *SIAM Review*, vol. 26 (1984), 351–366.

[38] J. M. Borwein and P. B. Borwein, *Pi and the AGM: A Study in Analytic Number Theory and Computational Complexity*, Canadian Mathematical Society Monographs, Wiley-Interscience, New York, 1987, reprinted 1998.

[39] J. M. Borwein, P. B. Borwein, and D. H. Bailey, "Ramanujan, modular equations and pi or how to compute a billion digits of pi," *American Mathematical Monthly*, vol. 96 (1989), 201–219; reprinted in *Organic Mathematics Proceedings*, http://www.cecm.sfu.ca/organics, April 12, 1996, with print version: *CMS/AMS Conference Proceedings*, vol. 20 (1997), ISSN: 0731–1036.

[40] J. M. Borwein and B. Salvy, "A proof of a recursion for Bessel moments," *Exp. Mathematics*, vol. 17 (2008), 223–230.

[41] J. M. Borwein and B. Sims, "The Douglas-Rachford algorithm in the absence of convexity," Chapter 6, pp. 93–109 in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering* in *Springer Optimization and Its Applications*, in press, 2011.

[42] J. M. Borwein, A. Straub, and J. Wan, "Three-step and four-step random walk integrals," *Experimental Mathematics*, to appear, Sept 2010, http://www.carma.newcastle.edu.au/~jb616/walks2.pdf.

[43] Jonathan M. Borwein, Armin Straub, James Wan and Wadim Zudilin, "Densities of short uniform random walks," Oct 2010, http://www.carma.newcastle.edu.au/~jb616/densities.pdf.

[44] R. P. Brent and P. Zimmermann, *Modern Computer Arithmetic*, Cambridge Univ. Press, 2010.

[45] H. W. Broer, C. Simó and R. Vitolo, "Chaos and quasi-periodicity in diffeomorphisms of the solid torus," *Discrete Contin. Dyn. Syst. Ser. B*, vol. 14 (2005), 871–905.

[46] C.W. Clenshaw, "A note on the summation of Chebyshey series," *Math. Tab. Wash.*, vol. 9 (1955) 118–120.

[47] G. Corliss and Y. F. Chang, "Solving ordinary differential equations using Taylor series," *ACM Trans. Math. Software*, vol. 8 (1982), 114–144.

[48] M. Czakon, "Tops from light quarks: Full mass dependence at two-Loops in QCD," *Phys. Lett. B*, vol. 664 (2008), 307, http://arxiv.org/abs/0803.1400.

[49] T.J. Dekker, "A floating-point technique for extending the available precision," *Numer. Math.*, vol. 18 (1971), 224–242.

[50] J. Demmel and P. Koev, "The accurate and efficient solution of a totally positive generalized Vandermonde linear system," *SIAM J. of Matrix Analysis Applications*, vol. 27 (2005), 145–152.

[51] W. D. Evans, L.L. Littlejohn, F. Marcellán, C. Markett and A. Ronveaux, "On recurrence relations for Sobolev orthogonal polynomials," *SIAM J. Math. Anal.*, vol. 26 (1995), 446–467.

[52] J. Dongarra, "LAPACK," http://www.netlib.org/lapack.

[53] J. Dongarra, "LINPACK," http://www.netlib.org/linpack.

[54] R. K. Ellis, W. T. Giele, Z. Kunszt, K. Melnikov and G. Zanderighi, "One-loop amplitudes for W+3 jet production in hadron collisions," manuscript, 15 Oct 2008, http://arXiv.org/abs/0810.2762.

[55] V. Elser, I. Rankenburg, and P. Thibault, "Searching with iterated maps", *Proceedings of the National Academy of Sciences*, vol. 104 (2007), 418–423.

[56] T. Ferris, *Coming of Age in the Milky Way*, HarperCollins, New York, 2003.

[57] A. M. Frolov and D. H. Bailey, "Highly accurate evaluation of the few-body auxiliary functions and four-body integrals," *J. Physics B*, vol. 36 (2003), 1857–1867.

[58] W. Gautschi, "Computational aspects of three-term recurrence relations," *SIAM Rev.*, vol. 9 (1967), 24–82.

[59] V. Gelfreich and C. Simó, "High-precision computations of divergent asymptotic series and homoclinic phenomena," *Discrete Contin. Dyn. Syst. Ser. B*, vol. 10 (2008), 511–536.

[60] S. Graillat, P. Langlois and N. Louvet, "Algorithms for accurate, validated and fast polynomial evaluation," *Japan J. Indust. Appl. Math.*, vol. 26 (2009), 191–214.

[61] S. Gravel, and V. Elser, "Divide and concur: A general approach to constraint satisfaction," *Phys. Rev. E*, vol. 78 (2008), 036706.

[62] C. Grebogi, E. Ott, S. Pelikan, and J. A. Yorke, "Strange attractors that are not chaotic," *Phys. D*, vol. 13 (1984), 261–268.

[63] E. Hairer, S. Nørsett and G. Wanner, *Solving ordinary differential equations. I. Nonstiff problems*, second edition, Springer Series in Computational Mathematics, vol. 8, Springer-Verlag, Berlin, 1993.

[64] Vincent Hakim and Kirone Mallick, "Exponentially small splitting of separatrices, matching in the complex plane and Borel summation," *Nonlinearity*, vol. 6 (1993), 57–70.

[65] A. Haro and C. Simó, "To be or not to be a SNA: That is the question," Preprint 2005-17 of the Barcelona UB-UPC Dynamical Systems Group (2005).

[66] P. H. Hauschildt and E. Baron, "The numerical solution of the expanding Stellar atmosphere problem," *J. Comp. and Applied Math.*, vol. 109 (1999), 41–63.

[67] W. Hayes, "Is the outer solar system chaotic?," *Nature Physics*, vol. 3 (2007), 689–691.

[68] Y. He and C. Ding, "Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications," *J. Supercomputing*, vol. 18 (Mar 2001), 259–277.

[69] Y. Hida, X. S. Li and D. H. Bailey, "Algorithms for Quad-Double Precision Floating Point Arithmetic," 15th IEEE Symposium on Computer Arithmetic (ARITH-15), 2001.

[70] Hao Jiang, Roberto Barrio, Housen Li, Xiangke Liao, Lizhi Cheng, Fang Su, "Accurate evaluation of a polynomial in Chebyshev form," *Applied Mathematics and Computation*, vol. 217 (2011), 9702–9716

[71] D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms.* Addison-Wesley, third edition, 1998.

[72] P. Koev, "Software," 2010, `http://math.mit.edu/~plamen/software`.

[73] G. Lake, T. Quinn and D. C. Richardson, "From Sir Isaac to the Sloan survey: Calculating the structure and chaos due to gravity in the universe," *Proc. of the 8th ACM-SIAM Symp. on Discrete Algorithms*, SIAM, Philadelphia, 1997, 1–10.

[74] J. S. W. Lamb, "Reversing symmetries in dynamical systems," *J. Phys. A: Math. Gen.*, vol. 25 (1992), 925–937.

[75] S. K. Lucas and H. A. Stone, "Evaluating infinite integrals involving Bessel functions of arbitrary order," *Journal of Computational and Applied Mathematics*, vol. 64 (1995), 217–231.

[76] V. F. Lazutkin, "Splitting of separatrices for the Chirikov standard map," *J. Math. Sci.*, vol. 128 (2005), 2687–2705.

[77] E. Lorenz, "Deterministic nonperiodic flow," *J. Atmospheric Sci.*, vol. 20 (1963), 130–141.

[78] J. E. Littlewood, *A Mathematician's Miscellany*, Methuen and Co., London, 1953, reprinted by Cambridge University Press, 1997.

[79] T. Ogita, S.M. Rump, and S. Oishi, "Accurate sum and dot product," *SIAM J. Sci. Comput.*, vol. 26 (2005), 1955–1988.

[80] G. Ossola, C. G. Papadopoulos and R. Pittau, "CutTools: A program implementing the OPP reduction method to compute one-loop amplitudes," *J. High-Energy Phys.*, vol. 0803 (2008), 042, `http://arxiv.org/abs/0711.3596`.

[81] W. H. Press, S. A. Eukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd edition, Cambridge University Press, 2007.

[82] S.M. Rump, "Verification methods: rigorous results using floating-point arithmetic," *Acta Numer.*, vol. 19 (2010), 287–449.

[83] Avram Sidi, "The numerical evaluation of very oscillatory infinite integrals by extrapolation," *Mathematics of Computation*, vol. 38 (1982), 517–529.

[84] Avram Sidi, "A user-friendly extrapolation method for oscillatory infinite integrals," *Mathematics of Computation*, vol. 51 (1988), 249–266.

[85] Avram Sidi, "A user-friendly extrapolation method for computing infinite-range integrals of products of oscillatory functions," *IMA Journal of Numerical Analysis*, to appear (2011) doi:10.1093/imanum/drr022.

[86] C. Simó, "Global dynamics and fast indicators," in *Global Analysis of Dynamical Systems*, 373–389, Inst. Phys., Bristol, 2001.

[87] H. Takahasi and M. Mori, "Double exponential formulas for numerical integration," *Pub. RIMS*, Kyoto University, vol. 9 (1974), 721–741.

[88] Tse-Wo Zse, personal communication to the authors, July 2010.

[89] D. Viswanath, "The Lindstedt-Poincaré technique as an algorithm for computing periodic orbits," *SIAM Review*, vol. 43 (2001), 478–495.

[90] D. Viswanath, "The fractal property of the Lorenz attractor," *Phys. D*, vol. 190 (2004), 115–128.

[91] D. Viswanath and S. Şahutŏglu, "Complex singularities and the Lorenz attractor," *SIAM Rev.*, vol. 52 (2010), 294–314.

[92] J. Wan, "Moments of products of elliptic integrals," preprint, October 2010.

[93] Z.-C. Yan and G. W. F. Drake, "Bethe logarithm and QED shift for Lithium," *Phys. Rev. Letters*, vol. 81 (2003), 774–777.

[94] T. Zhang, Z.-C. Yan and G. W. F. Drake, "QED corrections of $O(mc^2\alpha^7 \ln \alpha)$ to the fine structure splittings of Helium and He-Like ions," *Phys. Rev. Letters*, vol. 77 (1994), 1715–1718.